RSRE
MEMORANDUM No. 4160

ROYAL SIGNALS & RADAR ESTABLISHMENT

THE INSIDE/OUTSIDE ALGORITHM: GRAMMATICAL INFERENCE APPLIED TO STOCHASTIC CONTEXT-FREE GRAMMARS

Author: Lorraine Dodd

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

RSRE MEMORANDUM No. 4160

Royal Signals and Radar Establishment

Memorandum 4160

THE INSIDE-OUTSIDE ALGORITHM: GRAMMATICAL INFERENCE APPLIED TO STOCHASTIC CONTEXT-FREE GRAMMARS

Lorraine Dodd

June 1988

Abstract

This paper describes the Inside-Outside algorithm which re-estimates the rewrite rule probabilities of a stochastic context-free grammar. The particular example described in this memorandum is the application of the Inside-Outside algorithm to the spelling of English words.

Contents

A-1

List of Figures

List of Tables

1 Introduction

Current research at RSRE into language modelling for Automatic Speech Recognition (ASR) involves the study of formal methods of grammatical inference; in particular, the inference of stochastic context-free grammars. The Inside-Outside algorithm [2] re-estimates the rewrite rule probabilities of a stochastic context-free grammar from many examples of data strings (which may be words, sentences, sequences of grammatical tags, mathematical expressions, or anything that could be explained by a context-free grammar). The Inside-Outside algorithm is also known as Baker's algorithm as it is based on his 'nodal span' principle which generalises and extends the techniques used in Hidden Markov modelling [4] to stochastic context-free grammars. The crucial idea is that the hidden random variables are associated with spans (i.e. intervals or substrings of the data strings) rather than with single sample times as in the (finite-state) Hidden Markov models.

This paper describes the mathematics of the Inside-Outside algorithm, a procedure for programming the algorithm and some early results of grammatical inference work using simple context-free grammars and small structured samples of English words. Section 2 briefly introduces stochastic formal grammars (SFGs) and gives an example of a stochastic context-free grammar. (For a more detailed description of SFGs see [3].) Section 3 describes the mathematics of the Inside-Outside (I-O) algorithm. Section 4 briefly discusses the computer programming aspects. Section 5 describes some exploratory experiments using the I-O algorithm on simple structured word strings. Section 6 briefly shows how the inferred stochastic context-free grammar can be used to find the maximum likelihood parse of a string. The final section is a discussion about the problems of using the I-O algorithm for general grammatical inference.

2 Stochastic Formal Grammars

2.1 Introduction

A stochastic formal grammar (SFG) can be used to specify languages and to describe physical patterns and data structures. A SFG can also be used in a generative capacity. The rewrite rules of the SFG have associated probabilities which can be used in a random sampling process to generate, for example, letters to form a word string. The probabilities for all the rewrite rules with the same left hand side sum to unity. For example:

$$\begin{array}{lcll} S & \rightarrow & DC & 0.4 \\ S & \rightarrow & CD & 0.6 \\ D & \rightarrow & AB & 1.0 \\ C & \rightarrow & BA & 1.0 \\ B & \rightarrow & d & 0.3 \\ B & \rightarrow & l & 0.7 \\ A & \rightarrow & e & 0.6 \\ A & \rightarrow & a & 0.4 \end{array} \qquad (1)$$

The symbols C and D are non-terminal symbols which are rewritten according to the stochastic rules as AB or BA. (These rules will be called non-terminal rules.) The special (start) symbol S denotes the start of the rewrite process. Pre-terminal symbols are those non-terminal symbols which can only be rewritten as terminal symbols. The pre-terminal symbols are A and B. (The rules which rewrite pre- terminals will be called terminal rules.) The grammar will generate sequences of terminal symbols such as 'deal', 'lead', etc.

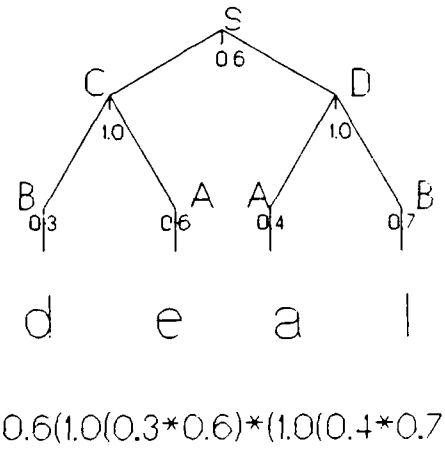

Figure 1: A derivation tree for the word 'deal'

(Capital italicised letters will be used to denote non-terminal symbols and lower case letters to denote terminal symbols.)

2.2 Tree Diagrams

A context-free grammar can be thought of as defining a set of tree diagrams. Each tree is labelled with a start symbol, S, at the root node, with terminal symbols at the leaves and non-terminal symbols labelling the inner nodes. Each node and its associated branches will be called a sub-tree and each sub-tree corresponds to a production rule in the grammar. Each tree has as its leaves a sequence of terminal symbols. Each tree has a probability associated with it which is the product of the probabilities associated with the sub-trees. The sum of all these tree probabilities is unity. Any particular sequence of terminal symbols may have more than one tree representation and the probability of the terminal string (given the grammar) is the sum of the probabilities of all these trees. The usual role of a tree diagram is to illustrate the parse (or derivation) of a word, sentence or data string according to the grammar.

For example, a derivation tree for the word 'deal' according to the grammar in (1) is as shown in Figure 1. The probability of the word 'deal' being generated by the grammar rules (which is shown in Figure 1) works out at about one chance in thirty-three (or 0.03).

2.3 Stochastic Context-free Grammars

If the rewrite rules of the grammar are of the form:

$$A \rightarrow \gamma^*$$

(* is the Kleene operator which denotes a sequence or repetition) where

$$A \epsilon V_N$$

and

$$\gamma \epsilon V_N \cup V_T$$

where V_N and V_T are the sets of non-terminal and terminal symbols, respectively. (that is, γ is either a non-terminal symbol or a terminal symbol and A is a non-terminal symbol), then the grammar is a context-free grammar (CFG).

In a context-free grammar any sequence of terminal and non-terminal symbols can appear on the right hand side of the rewrite rule but the left hand sides must consist of one non-terminal symbol only. Any CFG (even when stochastic) can be transformed into a more useful form (for the purposes of the I-O algorithm) called the Chomsky Normal Form (CNF) [1]:

$$\begin{aligned} A &\rightarrow BC \\ C &\rightarrow c \end{aligned}$$

Non-terminal rules in CNF have only two non-terminal symbols on their right hand side which limits the derivation tree to binary branches.

3 The Inside-Outside Algorithm

3.1 Introduction

The Inside-Outside (I-O) algorithm re-estimates the probabilities associated with each of the rules in a stochastic context-free grammar given many examples of terminal strings which have been (or could have been) generated by a SCFG. The I-O algorithm can only accept and work with rules which are written in CNF. Initially, either random probabilities or probabilities which represent some prior knowledge about the grammar can be assigned to initialise the algorithm. Strings from a sample training set are read into the algorithm and 'nodal span' probabilities are calculated. At the end of the training sample, the rewrite rule probabilities are updated and the process continues to the next iteration. In effect, the I-O algorithm considers all possible parses of the input string according to its current rewrite rule probabilities and counts the number of times each of the rules is used. After one pass through a training set, it normalises and weights these counts to give an estimate of the probability of each rule.

3.2 Terminology

The term 'interval' is used to refer to a substring of the input string and it is usually associated with a particular node in the tree diagram. (In Baker's terminology an interval is referred to as a 'span' but the term interval will be used in this paper to distinquish it from the use of 'span' as a verb.) Sub-tree probabilities are calculated for every interval (i.e from intervals of length one to the entire string) and for every node in the tree which could 'span' the interval. In following sections these sub-tree probabilities will be called the e and f values. For example, in Figure 2 the interval (delimited by square brackets) [de] has length two and the node that spans [de] is labelled C.

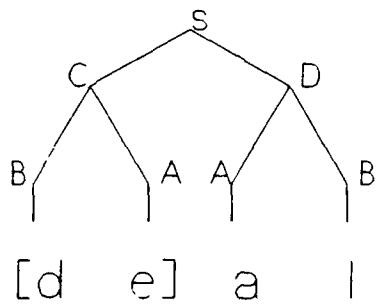

Figure 2: The interval [de] spanned by node labelled C

3.3 Notation

The rewrite rule probabilities fall into two distinct types. The probabilities which are associated with non-terminal rules will be called the A-matrix probabilities and the terminal rule probabilities will be called B-matrix probabilities.

The rewrite rule $label_i \rightarrow label_j\ label_k$, 1.0 is expressed in A-matrix notation as follows:

$$a_{ijk} = 1.0$$

A rewrite rule such as, $label_j \rightarrow terminal_k$, 0.5 is expressed as :

$$b_{jk} = 0.5$$

In terms of the derivation trees, the B-matrix probabilities are associated with the leaf (terminal) nodes and the A-matrix probabilities describe the statistics of the branches across the set of trees. The e and f values are denoted by $e[s,t,i]$ and $f[s,t,i]$ where s marks the first character of the interval, t marks the final character in the interval and i represents the labelled node in the tree which spans the interval.

3.4 The e and f values

Assume that there are L strings, $d_1, d_2, ..., d_l, ..., d_L$ in a training set. The lth string is $O_1^l....O_{T_l}^l$ and its length is T_l. For example, grammar (1) in Section 2.1 could generate as string, d_1, the word 'deal' so that T_1 is 4 and $O_1^1....O_4^1$ are 'd'..'l' respectively. The e-value, $e[s,t,i]$, is the probability of the interval $O_s...O_t$ given that the non-terminal, i, spans the interval. The f-value, $f[s,t,i]$, is the probability of the intervals $O_1...O_{s-1}$ and $O_{t+1}...O_T$ given that the non-terminal, i, spans the interval $O_s...O_t$. So the e-value gives the probability of the sub-trees 'inside' the span of the non-terminal and the f-value gives the probability of the sub-trees 'outside' the span of the non-terminal.

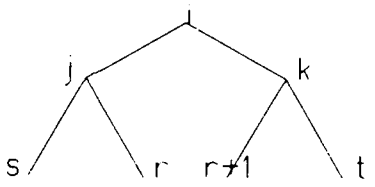

Figure 3: The calculation of $e[s,t,i]$

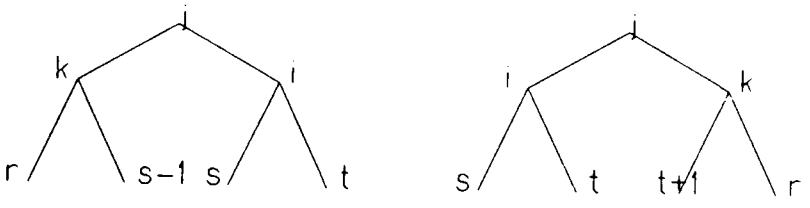

Figure 4: The calculation of $f[s,t,i]$

3.5 Calculation of the e and f values

The I-O algorithm is a kind of two-stage parsing process. The bottom up process generates the e values and effectively considers all possible configurations of the parse tree 'inside' the section below the node spanning the interval. (See Figure 3) The top down process generates the f values and effectively considers all possible labelled sub-trees 'outside' the section spanning the interval. (See Figure 4) For example, given that the algorithm is initialised with the probabilities of the SCFG as given in Section 2.1 and the input string is 'deal', then the basic steps in the I-O algorithm are as follows :

1. Assign the current B-matrix probabilities to the e values for the first stage in the bottom up process (i.e. for spans of length 1).

The only non-zero probabilities are :

$$\begin{aligned} e[1,1,B] &= b_{BO_1^1} = b_{Bd} = 0.3 \\ e[2,2,A] &= b_{AO_2^1} = b_{Ae} = 0.6 \\ e[3,3,A] &= b_{AO_3^1} = b_{Aa} = 0.4 \\ e[4,4,B] &= b_{BO_4^1} = b_{Bl} = 0.7 \end{aligned}$$

2. Now consider all possible parses of the intervals of length two using the e values, as assigned in step 1, and the existing A-matrix value for all possible nodes. The only non-zero probabilities are :

$$e[1,2,C] = e[1,1,B]\, e[2,2,A]\, a_{CBA}$$

and

$$e[3,4,D] = e[3,3,A]\, e[4,4,B]\, a_{DAB}$$

(Using this particular grammar there are no alternative trees for the word 'deal' and so there is only one term in the e-value probability calculation.)

3. Now consider the intervals of length three (and so on ..) so that all possible parses of the interval given the node are added into the e-value. The general equation for the e-value is :

$$e[s,t,i] = \sum_{r,j,k} e[s,r,j]\, e[(r+1),t,k]\, a_{ijk}$$

where r varies between s and t.

4. The final step in the bottom up process calculates the e-value for the entire string. This e-value, $e[1,T_1,S]$, is the probability of the word 'deal' given the current SCFG (i.e. the current A-matrix and B-matrix).

5. The first step in the top down process is to set $f(1,T_1,S) = 1.0$.

6. The f values are then calculated using the f values which were calculated in previous steps of the top down process and the e values as appropriate to the portions of the string outside the interval$[s..t]$. The general equation for the f values is :

$$f[s,t,i] = \sum_{r,j,k} f[r,t,j] e[r,s-1,k] a_{jki} + \sum_{r,j,k} f[s,r,j] e[t+1,r,k] a_{jik}$$

where in the first term r varies between 1 and $s-1$ (i.e. for the substring on the left hand

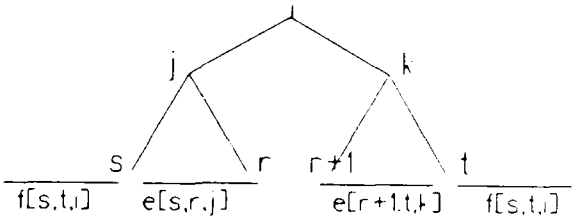

Figure 5: Calculation of the weights

side of $[s..t]$) and in the second term r varies between $t+1$ and T_1 (i.e. for the substring on the right hand side of $[s..t]$).

The I-O algorithm for SCFGs is analogous to the Forward/Backward algorithm for Hidden Markov models [4] and uses the same kind of iterative Dynamic Programming technique. The α and β in the Forward/Backward algorithm are the equivalent of the e and f values, respectively.

3.6 The Re-estimation Process

Before describing the re-estimation process it is useful to define some extra notation so that the equations simply become a summation of weights :

$$\begin{aligned} w_{stijk} &= \sum_r f[s,t,i]\, e[s,r,j]\, e[r+1,t,k]\, a_{ijk} \\ w_{sti} &= e[s,t,i]\, f[s,t,i] \quad (= \textstyle\sum_{jk} w_{stijk}) \end{aligned}$$

The weight, w_{sti}, is the probabilty of the string $O_1 \ldots O_T$ given that the interval $O_s \ldots O_t$ is spanned by non-terminal, i. The weight, w_{stijk}, is the probability of the string $O_1 \ldots O_T$ given that the interval $O_s \ldots O_t$ is spanned by non-terminal, i and that the non-terminal, i, rewrites as non-terminals, j and k. (See Figure 5.)

The A-matrix probabilities are updated as follows :-

$$\bar{a}_{ijk} = \frac{\sum_{l=1}^{L} p_l^{-1} \sum_{st} w^l_{stijk}}{\sum_{l=1}^{L} p_l^{-1} \sum_{st} w^l_{sti}} \tag{2}$$

where p_l is the probability of the lth string according to the current probabilities and it is represented in the algorithm as $e[1, T_l, S]$ (the l subscript represents the string index).

So the 'inside' and 'outside' parsing probabilities (i.e. the e and f values) are combined with the current rewrite probability for each binary branch and this is normalised by dividing by the sum of all possible binary branches to give the new A- matrix values. The probability of the string, p_l, is used to weight the statistics in favour of the more unlikely strings to prevent the rule probabilities 'feeding' only from the more common strings as the iterative process continues.

Frequency counts of the strings can be used quite simply in the re-estimation equations by multiplying the weight summation. Denoting the frequency of the lth string by $freq_l$ the re- estimation equation becomes :

$$\bar{a}_{s,y,k} = \frac{\sum_{l=1}^{L} freq_l \; p_l^{-1} \sum_{st} u^l_{st(y)k}}{\sum_{l=1}^{L} freq_l \; p_l^{-1} \sum_{st} u^l_{st_1}} \tag{3}$$

The B-matrix probabilities are updated similarly :

$$\bar{b}_{yk} = \frac{\sum_{l=1}^{L} p_l^{-1} \sum_{t \; O_t^l = k} u^l_{tt_j}}{\sum_{l=1}^{L} p_l^{-1} \sum_{t} u^l_{tt_j}} \tag{4}$$

so that the numerator only contains those weights for which the entry at position t in the string is equal to terminal k. The string frequency can be used as in equation (3) :

$$\bar{b}_{yk} = \frac{\sum_{l=1}^{L} freq_l \; p_l^{-1} \sum_{t \; O_t^l = k} u^l_{tt_j}}{\sum_{l=1}^{L} freq_l \; p_l^{-1} \sum_{t} u^l_{tt_j}} \tag{5}$$

4 The Computer Program

4.1 Introduction

The Inside-Outside algorithm computer program is written in modular form in VAX PASCAL. Details of how to run the program and the I-O algorithm demonstration are given in the Appendix. The following sections describe each of the modules.

4.2 The Main Program

The main program simply reads in all the control parameters (such as the number of non-terminal labels, etc and the various file names). It then performs the main iteration loop for the required number of training cycles.

4.3 Initialisation

This small module simply initialises the weights to zero at the beginning of each iteration

4.4 Probability Assignment

The A-matrix and B-matrix probability values are initialised according to the user's choice from the following three options:

- random numbers between 0 and 1.
- uniform values

- probabilities taken from a file which represent prior knowledge about the grammar or from a previous run.

All probabilties are then normalised to conform with the conditions for a SCFG. In the case of the use of prior knowledge about a particular grammar the probabilities in the 'prior knowledge' file are slightly reduced due to all other (i.e. unspecified) matrix probabilities being initialised as 'a small number' to avoid initialising probabilities as zero. (If probabilities have zero value then they remain at zero due to the multiplicative functions in the algorithm.)

4.5 Calculation of e and f values

Before each iteration the e and f values are initialised to zero. The values are then simply accumulated over the training set. The bottom up process is controlled using the following loops :

```
interval length goes from 1 up to string length

s goes from 1 to string length-span length+1
```

The top down process is controlled using the following loops :

```
interval length goes from string length down to 1

t goes from string length to span length
```

The variable r moves between s and t and the i, j and k nodes assume the values of all possible labels up to the limits specified for the non- and pre-terminals.

4.6 Matrix Update

At the end of each iteration the e and f values are used to update the A-matrix and B-matrix probabilities as shown in equations (3) and (5). After the required number of iterations, the values of the A-matrix and B-matrix and their corresponding indices are output as the inferred grammar rules.

4.7 Notes

Pre-terminals are given arbitrary labels in the I-O algorithm but they must be treated quite distinctly from other non-terminal symbols. Non-terminal symbols are also given arbitrary names but a special non- terminal symbol must designated to be the root (start) symbol, S. Binary CNF rules (e.g. $S \rightarrow AB$) are assumed throughout.

5 The Experiments

This section describes some of the exploratory work which was carried out to validate and verify the computer programs and to examine the general behaviour of the algorithm.

5.1 The Input Data

In order to provide training data for the algorithm, the computer readable LOB (Lancaster-Oslo-Bergen) corpus of English words was used, and all words marked in the corpus as proper nouns and foreign words were excluded. Hyphenated words and those containing apostophes were included and the hyphen and apostrophe are treated as alphabetic characters. Several small subsets were selected from the corpus for the purposes of testing and training. In the example described in 5.2, a typical training set is a subset of words which all contain the substring 'ght':

1	*brighter*	2	*bright*
3	*bought*	3	*caught*
6	*delight*	2	*eight*
4	*flight*	3	*height*
1	*insight*	6	*fight*
1	*lights*	13	*brought*
3	*caught*	5	*daughter*
2	*fighter*	2	*flights*
3	*height*	5	*light*
1	*nights*	1	*ought*
3	*rights*	1	*sights*
1	*sought*	14	*thought*
3	*tonight*	1	*upright*
4	*weight*	31	*night*
17	*right*	2	*sight*
3	*tonight*	1	*upright*
4	*weight*	31	*night*
3	*rights*	2	*slight*
2	*straight*	1	*taught*
1	*upright*	4	*weight*

where the number before each word is a frequency count of the number of times that the word occurs in the sample taken from the corpus.

5.2 A Grammar for spelling -ght- words

The algorithm starts-out with random rewrite rule probabilities which are correctly normalised. The algorithm also needs to be told how many non-terminal and pre-terminal symbols it is to use for its SCFG. The e and f values are calculated and accumulated for every word in the training set and at the end of each training set the A-matrix and B-matrix are updated.

The next iteration uses the new A-matrix and B-matrix probabilities to calculate the e and f values. In this way the algorithm re-estimates the probabilities according to the structure in the words of the training set.

5.3 Results

The algorithm uses input data similar to the lists in 5.1 and the number of labels for non-terminal and pre- terminal symbols is varied in order to examine the effects of changing these numbers on the resulting grammar. The number of matrix updates is also varied to discover the rate of change of the grammar as the algorithm is exposed to more training

Results on -ght- words		
updates	non,pre-terminals	Aver word prob
20	20,8	0.0001020
20	24,12	0.0000802
30	20,8	0.0069200
30	24,10	0.0133443
30	24,12	0.0213200
30	22,10	0.0216900

Table 1: Results of grammatical inference for -ght- words

data. The results in Table 5.3 were obtained and are listed in ascending order of probability value.

The first column shows the number of matrix updates (i.e. the number of training set iterations) before the grammatical inference process is stopped. The second column gives the upper limit on the number of non- terminal and pre-terminal symbol labels that the algorithm uses. The third column shows the average word probability taken over all the words in the training set given the SCFG that the algorithm has inferred in its final iteration. Figure 6 shows how the inferred grammars gradually get better at explaining the data in the training sets. The grammar 'score' is the logarithm of the average word probability at the end of each iteration. It takes several iterations for the node labels to become organised and then the grammar score rapidly increases. The rate of increase slows down until further iterations produce only small improvements.

The inferred SCFG can be used in a simple random sampling process to generate synthetic 'word strings'. The following list of words was generated from the grammar which resulted from the run with 30 updates, 22 non-terminals and 10 pre-terminals :

night	*pright*	*light*	*rright*
fenight	*aight*	*enight*	*light*
baight	*night*	*sight*	*right*
eight	*light*	*night*	*wwnight*
night	*night*	*enight*	*night*
eight	*night*	*eight*	*rught*
sright	*night*	*tnight*	*night*
light	*aight*	*rught*	*aight*
elught	*night*	*eight*	*enight*
cnight	*night*	*night*	*unnight*
night	*night*	*ulight*	*night*
lught	*night*	*night*	*lught*
fecfnight	*wnight*		

6 A Maximum Likelihood Parser

6.1 Introduction

The inferred SCFG can be used to generate the parse of an input word string which gives a maximum likelihood value. (This is called the maximum likelihood parse.) The e values are calculated in the normal bottom up process but the maximum e-value is stored for each

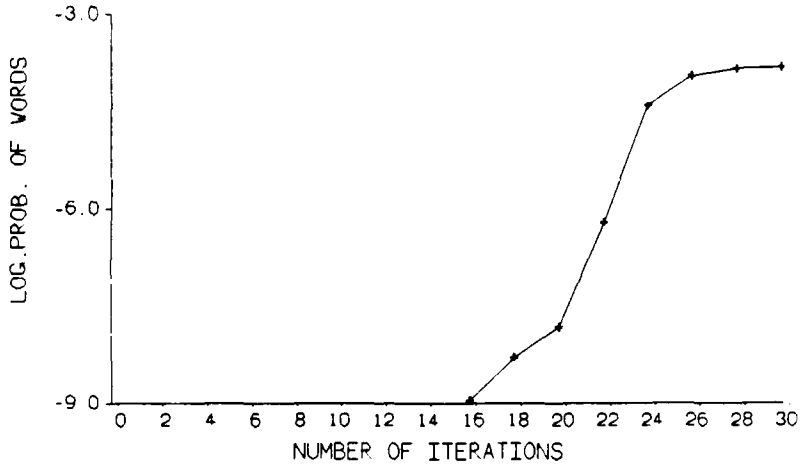

Figure 6: Grammar 'score' against number of iterations

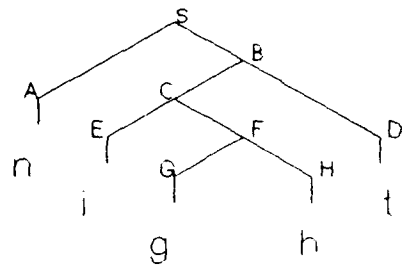

Figure 7: Maximum likelihood parse tree for the word 'night'

node label and interval. Pointers are set to the corresponding j and k (and r) so that in the top down process the tree can be traversed to generate the maximum likelihood parse. The process is basically one of dynamic programming; each interval (or span) is a stage and the non-terminal node labels are the states over which maximisation occurs at each stage. The recursive procedure calculates the maximum likelihood parse at each stage by taking into account the maximum likelihood parses at previous stages (i.e. for increasing size of interval). The final stage is reached at the root node (which spans the whole input string) where the value of the recursive maximisation function is the maximum likelihood of the string and the pointers which were set at each stage can be traced back down the tree to give the parse.

As an example, the maximum likelihood parse of the word 'night' is as shown in Figure 7 for the grammar inferred from 30 updates of -ght training sets with 22 non- terminals and 10 pre-terminals. (The node labels are those used within the program.)

6.2 Parsing using simple word tags

As a simple extension of the word-spelling experiments the sequences of single letter word tags from the LOB corpus are analysed using the I-O algorithm. Sentences of length less than six words are extrected from the LOB corpus and the first letter of the tag for each word in the sentence is written into a string which represents the broad part of speech analysis of the sentence. For example, the sentence 'The cat sat on the mat.' would be represented as 'ANVIAN' (i.e. article,noun,verb,preposition,article,noun). Such strings are used as training data for the algorithm. The inferred grammar rules could then be used to find the maximum likelihood parse of test sentences such as the one in Figure 6.2.

7 Discussion

The results of the simple -ght- word experiments show that too many non-terminal node labels can give lower grammar scores due to the extra processing required to organise the greater number of symbols.

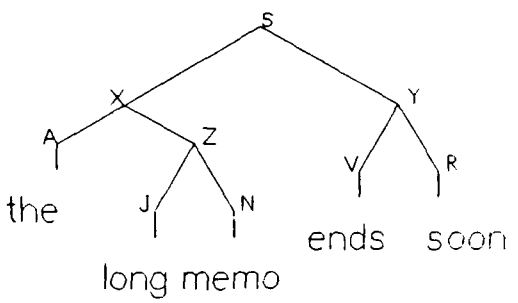

Figure 8: Maximum likelihood parse tree for the tag string AJNVR

A serious problem with the I-O algorithm is one of 'underflow'. In other words, the e and f value calculations involve multiplications of (often) very small probabilities which quickly approach the lower limit for a real value in computer terms. So far, the small sample training sets have not caused any underflow problems but future work (which is now being carried out) on general English spelling is more likely to run into underflow problems so, to get around the problem, the e and f calculations and matrix re-estimations make use of logarithms (and so addition rather than multiplication).

The algorithm is processor intensive (in particular, the e and f calclulations) in terms of computing time for each iteration. The processor time scales with the number of non-terminals and terminals, the number of iterations and the length of the input strings; so to infer grammars which have many symbols and which can generate long strings, the A-matrix and B-matrix would have to be initialised with probabilities which reflect prior knowledge about the grammar (rather than random numbers), in order to reduce the number of iterations necessary to achieve a grammar with a high score (see Figure 6). The algorithm is well suited to parallel processing, however, as the e and f values could be calculated quite separately for each string in the training set. Alteratively, or in addition, parallel or vector processors couls be used for the e and f calculations by exploiting their structure, which is basically that of matrix multiplication.

8 Appendix

The I-O algorithm program is in [DODD.IO] and the following modules need to be linked :

- BAKERMAIN
- ASSIGN
- INIT WTS
- EFCALC
- UPDATE MATRICES
- AEDMOD
- AEDLIB (LIBRARY)

The VMS command to link the above modules is called LINKBAKER.COM and it is executed by typing @LINKBAKER. BAKERTEST.COM runs a test version of the algorithm and an example of the control data for the perticular test run is given below:

```
set def [dodd.io]
run bakermain
random (* other options are 'prior knowledge' and 'uniform' *)
baker.out (* name of general output file *)
genfil.inp (* name of file for final inferred grammar rules *)
prifil2.inp (* prior knowledge rule file - if used *)
matfil.out (* file for final a-matrix and b-matrix *)
3895 (* initial random number seed *)
3 (* number of non-terminals *)
2 (* number of pre-terminals *)
1 (* number of training sets *)
N (* selection Y/N for uppercase/lowercase input *)
N (* Y/N for spaces between terminals in input strings *)
Y (* Y/N for AED display *)
aedfil2.inp (* file for AED display *)
expert (* control data for the colour routines *)
x=10,y=510,show
n=0,rgb=0,0,100,j
n=1,rgb=3*0,j
```

```
n=255,sc

exit (* control instructions for AED display *)

abc.inp (* name of file for training set *)

0.01 (* cut off value for output to genfil *)
```

If an AED dispay is not required then the AEDfilename and the control data for the AED displays should not be included in the command file.

References

[1] A.V. Aho and J.J. Ullman. *The Theory of Parsing, Translation and Compiling: Parsing* Prentice-Hall, 1972.

[2] J.K. Baker. Trainable grammars for speech recognition. In D.H.Klatt and J.J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustic Society of America*, pages 547–550, 1979.

[3] J.S. Bridle and L. Dodd. *Formal Grammars and Markov Models*. Memorandum 4051, RSRE, October 1987.

[4] F. Jelinek. Markov source modelling of text generation. In *NATO Advanced Study Institute: Impact of Processing Techniques on Communication*, Martinus Nijhoff, 1985

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	Memo 4160		U/C

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
778400	ROYAL SIGNALS & RADAR ESTABLISHMENT ST ANDREWS ROAD, GREAT MALVERN, WORCESTERSHIRE WR14 3PS
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title Inside - Outside algorithm: grammatical inference applied to Stochastic context-free grammars

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
Dodd, L			1988.6	17

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

UNLIMITED

Descriptors (or keywords)

continue on separate piece of paper

Abstract

This paper describes the Inside-Outside algorithm which re-estimates the re-write rule probabilities of a stochastic context-free grammar. The particular example described in this memorandum is the application of the Inside-Outside algorithm to the spelling of English words.